Mapping Reducibility

Reading

Section 5.3

Practice problems (page 211): 5.2, 5.4, 5.5, 5.6, 5.7, 5.9, 5.13, 5.22, 5.23, 5.28

Challenge: 5.16, 5.24

Mapping Reducibility

Mapping reducibility is a specific formulation of the generic idea of reducibility, revolving around the idea of functions, or "maps":

> A function $f\colon \Sigma^* \to \Sigma^*$ is called a **computable function** if there is a some Turing Machine M that on each input w halts with just $f(w)$ on the tape.

You can think of most arithmetic operations for starters as computable functions, whose input consists of an appropriate concatenation of the two numbers, and whose output is the sum of the two numbers (and if the input is of another form then the result doesn't matter). The computable functions we will mostly consider take as inputs the string representations of Turing Machines and return the string representations of other Turing Machines that have a desired behavior.

Note that we do not care about the state that the Turing Machine ends up at, only that it halts and does not loop.

Now we can define mapping reducibility:

> A language A is said to be **mapping reducible** to a langauge B, and we write $A \leq_m B$, if there is a computable function $f\colon \Sigma^* \to \Sigma^*$ where for every w we have:
>
> $$w \in A \text{ if and only if } f(w) \in B$$
>
> We then say that f is a **reduction** of A to B.
>
> An important consequence of this definition is that if $A \leq_m B$ then we also have $\bar{A} \leq_m \bar{B}$.

Let us look at an example of this: In our recent proofs we built the following mapping reduction:

$$f\colon A_{\text{TM}} \to E_{\text{TM}}$$

as follows:

If $w = \langle M, w\rangle$ is the representation of a TM and a string for it, then $f(w) = \langle N\rangle$ where N is the Turing Machine that on input x rejects unless $x = w$ and in that case runs M on w.

If w is not of this special form, then we assign to $f(w)$ some value that is not in E_{TM}, say for instance the empty string.

We now have to show that $w \in A_{\text{TM}}$ if and only if $f(w) \in E_{\text{TM}}$. This is in fact exactly what we did in the proof that E_{TM} is undecidable.

In fact most of the proofs we saw previously are examples of mapping reductions.

A lot of our previous proofs can be formalized in the following:

> If $A \leq_m B$, then:
>
> - If B is decidable, then A is decidable
> - If A is undecidable, then B is undecidable

The proof is straightforward: Suppose that D is a decider for B. We build a decider for A as follows:

- Suppose f is the reduction of A to B.
- Given an input w, compute $f(w)$.
- Now run D on $f(w)$ and return the result.
- This is a decider for A, since D accepts $f(w)$ if and only if $f(w) \in B$ if and only if $w \in A$.

Mapping reduction also works for Turing-recognizability. The same proof above suffices.

> If $A \leq_m B$, then:
>
> - If B is Turing-recognizable, then A is Turing-recognizable
> - If A is not Turing-recognizable, then B is not Turing-recognizable

We are going to use this approach to prove the following:

> EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

We prove this via two mapping reductions:

1. $A_{\text{TM}} \leq_m \overline{\text{EQ}_{\text{TM}}}$
2. $A_{\text{TM}} \leq_m \text{EQ}_{\text{TM}}$

By thinking of the complements in both of those reductions, the frist one tells us that EQ_{TM} is not Turing-recognizable, while the second tells us that $\overline{\text{EQ}_{\text{TM}}}$ is not Turing-recognizable.

Let us start with the first mapping reduction:

- We need to build a computable function f that on input $\langle M, w\rangle$ will produce a pair of machines so that M accepting w should be equivalent to the two machines not recognizing the same language.
- Here is how we will do that: M_1 will be the Machine that rejects all input.
- M_2 is the machine that on any input runs M on w and returns the result.
- This pair $\langle M_1, M_2\rangle$ is the result of f.

So M_2 is either the empty language (hence equal to M_1), if M doesn't accept w, or it is all strings (hence not equal to M_1) if M accepts w.

So $\langle M, w\rangle \in A_{\text{TM}}$ if and only if $f(\langle M, w\rangle) \in \overline{\text{EQ}_{\text{TM}}}$

The second mapping reduction is very similar: use as M_1 a Turing-Machine that accepts all input.